OpenCMISS-iron examples and tests used by *OpenCMISS* developers at University of Stuttgart, Germany

Christian Bleiler,* Dr.-Ing. Nehzat Emamy,†
Andreas Hessenthaler,* Thomas Klotz,*
Aaron Krämer,‡ Benjamin Maier,† Sergio Morales,*
Mylena Mordhorst,* Harry Saini*

July 25, 2017
18:33

CONTENTS

* Institute of Applied Mechanics (CE), University of Stuttgart, Pfaffenwaldring 7, 70569 Stuttgart, Germany
† Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
‡ Lehrstuhl Mathematische Methoden für komplexe Simulation der Naturwissenschaft und Technik, University of Stuttgart, Allmandring 5b, 70569 Stuttgart, Germany

LIST OF FIGURES

LIST OF TABLES

1   INTRODUCTION

This document contains information about examples used for testing *OpenCMISS-iron*. Read: How-to[1] and [1].

1.1   Cmgui files for cmgui–2.9

1.2   Variations to consider

- Geometry and topology

    1D, 2D, 3D

    Length, width, height

    Number of elements

    Interpolation order

    Generated or user meshes

    quad/hex or tri/tet meshes

- Initial conditions

- Load cases

    Dirichlet BC

    Neumann BC

    Volume force

    Mix of previous items

- Sources, sinks

- Time dependence

    Static

    Quasi-static

    Dynamic

- Material laws

    Linear

    Nonlinear (Mooney-Rivlin, Neo-Hookean, Ogden, etc.)

    Active (Stress, strain)

- Material parameters, anisotropy

- Solver

    Direct

    Iterative

- Test cases

    Numerical reference data

    Analytical solution

- A mix of previous items

---

1 https://bitbucket.org/hessenthaler/opencmiss-howto

1.3 Folder structure

TBD..

2 PROGRESS

People working on setting up tests in alphabetical order (surnames) with initials:

- CB : Christian Bleiler

- NE : Dr.-Ing. Nehzat Emamy

- AH : Andreas Hessenthaler

- TK : Thomas Klotz

- AK : Aaron Krämer

- BM : Benjamin Maier

- SM : Sergio Morales

- MM : Mylena Mordhorst

- HS : Harry Saini

2.1 Equations to test

Test single-physics problems before multi-physics problems!

- Diffusion equation (Laplace, Poisson, Generalized Laplace, ALE Diffusion, etc.)

- Linear elasticity equation (compressible and incompressible)

- Finite elasticity equation (compressible and incompressible Mooney-Rivlin, etc.)

- Navier-Stokes equation (ALE, Stokes, etc.)

- Monodomain equation

- CellML models

- Skeletal muscle models

- Fluid-structure interaction

- etc.

2.2 Setting up a new test

Use the following guideline to set up a new test:

1. Check if it is already there

2. Talk to other developers

3. Create a new subfolder examples/example-0xxx

4. Document the setup (computational domain, etc.) in examples/example-0xxx/doc/example.tex

5. Set up example with all parameters as command line arguments, see Section 1.2

6. Set up reference results (CHeart, Abaqus, analytical solution, etc.)

7. Set up script to run all tests in your example directory

8. Set up script to perform comparison between iron results and reference results

9. Set up visualization scripts

10. Compile, run, test, visualize your example

11. Compile, run, test, visualize all examples

For each example, progress is documented in the respective section titles with the following TAG:

- DOCUMENTED: finish the documentation of the example (spatial domain, number of time steps, boundary conditions, etc.

- COMPILES: example compiles (for default parameters)

- RUNS: example runs (for default parameters)

- CONVERGES: no convergence issues (for default parameters, results not plausible)

- PLAUSIBLE: results look sensible (for default parameters)

- VALIDATED: for all parameter sets it gives the correct results as compared to CHeart/Abaqus/analytical solution (includes visualization scripts, run scripts, comparison scripts, documentation!, . . .)

Move all tags CONVERGE, PLAUSIBLE to VALIDATED.

Next steps include:

- Everybody runs everything!

- Meeting with Oliver

- Meeting with Auckland

2.3 Long–term goals

- Different testing targets

  SMALL : small, fast tests

  BIG : same as before; further, bigger and more complex geometries, convergence analysis

  PARALLEL : same as before but in parallel

- Add more examples/those which were on the agenda but not started

- Jenkins continuous testing, integration and deployment

  test SMALL/BIG/PARALLEL targets

  integrate with GitHub (pull-requests triggers Jenkins, merge on success)

3   DIFFUSION EQUATION

3.1   Equation in general form

The governing equation is,

$$\partial_t u + \nabla \cdot [\sigma \nabla u] = f, \tag{1}$$

with conductivity tensor σ. The conductivity tensor is,

- defined in material coordinates (fibre direction),
- diagonal,
- defined per element.

3.2   Example–0001 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

3.2.1   *Mathematical model – 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad\qquad \Omega = [0, 2] \times [0, 1], \qquad\qquad (2)$$

with boundary conditions

$$u = 0 \qquad\qquad x = y = 0, \qquad\qquad (3)$$
$$u = 1 \qquad\qquad x = 2, y = 1. \qquad\qquad (4)$$

No material parameters to specify.

3.2.2   *Mathematical model – 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad\qquad \Omega = [0, 2] \times [0, 1] \times [0, 1], \qquad\qquad (5)$$

with boundary conditions

$$u = 0 \qquad\qquad x = y = z = 0, \qquad\qquad (6)$$
$$u = 1 \qquad\qquad x = 2, y = z = 1. \qquad\qquad (7)$$

No material parameters to specify.

3.2.3   *Computational model*

- Commandline arguments are:

  float: length along x-direction

  float: length along y-direction

  float: length along z-direction (set to zero for 2D)

  integer: number of elements in x-direction

  integer: number of elements in y-direction

  integer: number of elements in z-direction (set to zero for 2D)

  integer: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0

  2.0 1.0 0.0 4 2 0 1 0

  2.0 1.0 0.0 8 4 0 1 0

  2.0 1.0 0.0 2 1 0 2 0

  2.0 1.0 0.0 4 2 0 2 0

  2.0 1.0 0.0 8 4 0 2 0

  2.0 1.0 0.0 2 1 0 1 1

  2.0 1.0 0.0 4 2 0 1 1

```
2.0 1.0 0.0 8 4 0 1 1
2.0 1.0 0.0 2 1 0 2 1
2.0 1.0 0.0 4 2 0 2 1
2.0 1.0 0.0 8 4 0 2 1
2.0 1.0 1.0 2 1 1 1 0
2.0 1.0 1.0 4 2 2 1 0
2.0 1.0 1.0 8 4 4 1 0
2.0 1.0 1.0 2 1 1 2 0
2.0 1.0 1.0 4 2 2 2 0
2.0 1.0 1.0 8 4 4 2 0
2.0 1.0 1.0 2 1 1 1 1
2.0 1.0 1.0 4 2 2 1 1
2.0 1.0 1.0 8 4 4 1 1
2.0 1.0 1.0 2 1 1 2 1
2.0 1.0 1.0 4 2 2 2 1
2.0 1.0 1.0 8 4 4 2 1
```

3.2.4 Result summary

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 24 / 24
```

```
No failed tests.
```


Figure 1: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

Figure 2: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

Figure 3: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

Figure 4: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

3.3 Example–0001–u [PLAUSIBLE]

Example uses user-defined regular meshes in CHeart mesh format and solves a static problem, i.e., applies the boundary conditions in one step.

Issues: Interpolation type 8 (quadratic simplex) gives significantly different results compared to CHeart whereas all other simplex mesh results correspond rather well..

3.3.1 *Mathematical model – 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \qquad \Omega = [0, 2] \times [0, 1], \qquad \qquad (8)$$

with boundary conditions

$$u = 0 \qquad \qquad x = y = 0, \qquad \qquad (9)$$
$$u = 1 \qquad \qquad x = 2, y = 1. \qquad \qquad (10)$$

No material parameters to specify.

3.3.2 *Mathematical model – 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \qquad \Omega = [0, 2] \times [0, 1] \times [0, 1], \qquad \qquad (11)$$

with boundary conditions

$$u = 0 \qquad \qquad x = y = z = 0, \qquad \qquad (12)$$
$$u = 1 \qquad \qquad x = 2, y = z = 1. \qquad \qquad (13)$$

No material parameters to specify.

3.3.3 *Computational model*

- Commandline arguments are:

    float: length along x-direction

    float: length along y-direction

    float: length along z-direction (set to zero for 2D)

    integer: number of elements in x-direction

    integer: number of elements in y-direction

    integer: number of elements in z-direction (set to zero for 2D)

    integer: interpolation order (1: linear; 2: quadratic)

    integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

    2.0 1.0 0.0 2 1 0 1 0

    2.0 1.0 0.0 4 2 0 1 0

    2.0 1.0 0.0 8 4 0 1 0

    2.0 1.0 0.0 2 1 0 2 0

```
2.0 1.0 0.0 4 2 0 2 0
2.0 1.0 0.0 8 4 0 2 0
2.0 1.0 0.0 8 4 0 7 0
2.0 1.0 0.0 8 4 0 8 0
2.0 1.0 0.0 2 1 0 1 1
2.0 1.0 0.0 4 2 0 1 1
2.0 1.0 0.0 8 4 0 1 1
2.0 1.0 0.0 2 1 0 2 1
2.0 1.0 0.0 4 2 0 2 1
2.0 1.0 0.0 8 4 0 2 1
2.0 1.0 0.0 8 4 0 7 1
2.0 1.0 0.0 8 4 0 8 1
2.0 1.0 1.0 2 1 1 1 0
2.0 1.0 1.0 4 2 2 1 0
2.0 1.0 1.0 8 4 4 1 0
2.0 1.0 1.0 2 1 1 2 0
2.0 1.0 1.0 4 2 2 2 0
2.0 1.0 1.0 8 4 4 2 0
2.0 1.0 1.0 8 4 4 7 0
2.0 1.0 1.0 8 4 4 8 0
2.0 1.0 1.0 2 1 1 1 1
2.0 1.0 1.0 4 2 2 1 1
2.0 1.0 1.0 8 4 4 1 1
2.0 1.0 1.0 2 1 1 2 1
2.0 1.0 1.0 4 2 2 2 1
2.0 1.0 1.0 8 4 4 2 1
2.0 1.0 1.0 8 4 4 7 1
2.0 1.0 1.0 8 4 4 8 1
```

- Note: Binary uses command line arguments to search for the relevant mesh files.

3.3.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 30 / 32

current_run/l2x1x0_n8x4x0_i8_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 0.05804
current_run/l2x1x0_n8x4x0_i8_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 0.05804
```

Figure 5: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

Figure 6: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

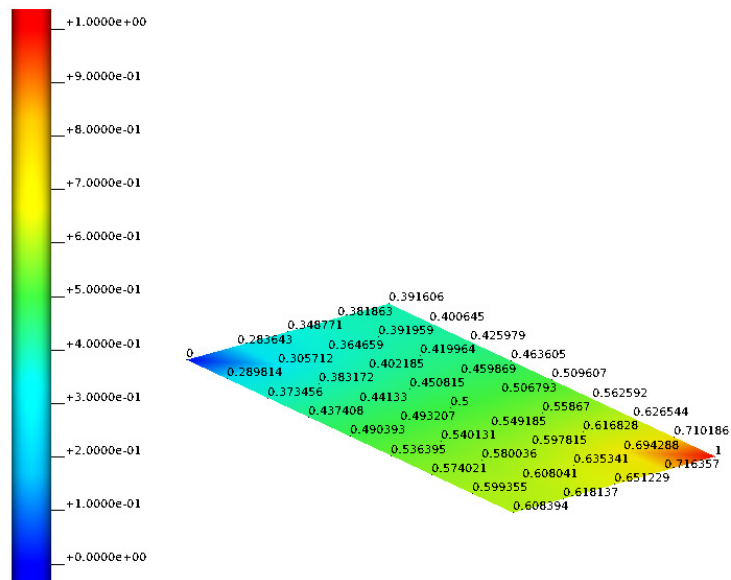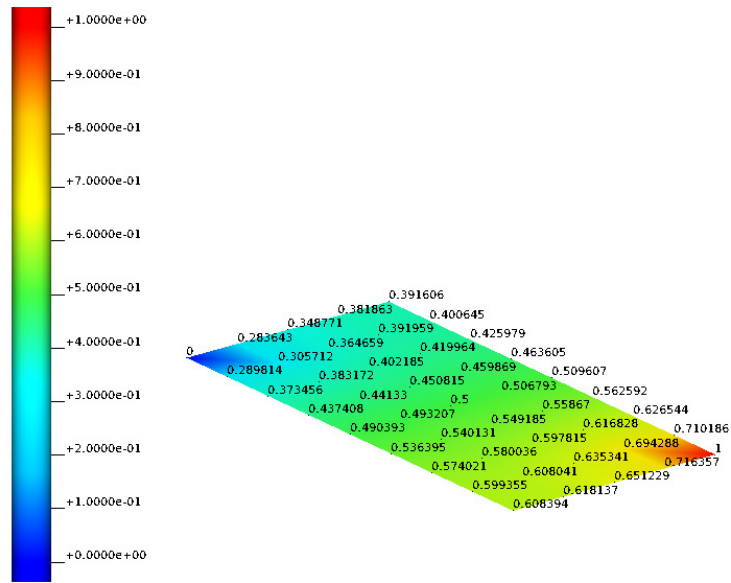**Figure 7:** 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

Figure 8: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].
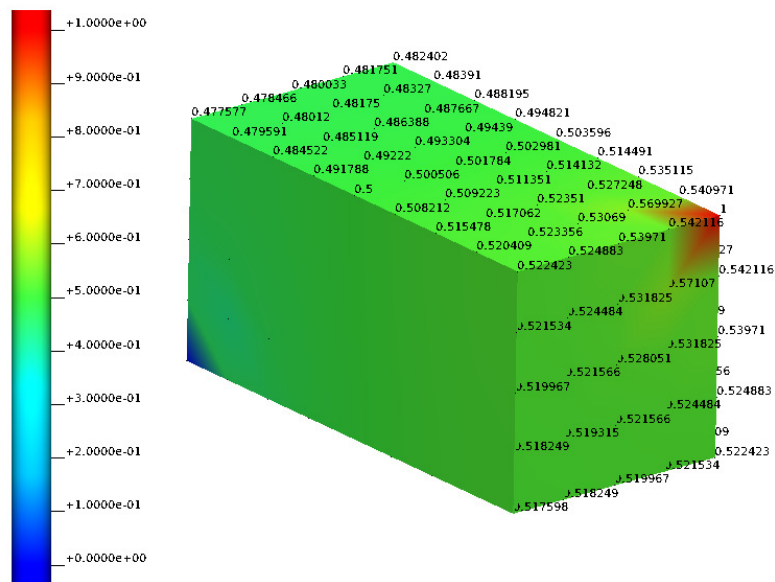
3.4 Example–0002 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

3.4.1 *Mathematical model – 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \qquad \Omega = [0,2] \times [0,1], \qquad (14)$$

with boundary conditions

$$u = 15y \qquad \qquad x = 0, \qquad (15)$$
$$u = 25 - 18y \qquad \qquad x = 2. \qquad (16)$$

No material parameters to specify.

3.4.2 *Mathematical model – 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad \qquad \Omega = [0,2] \times [0,1] \times [0,1], \qquad (17)$$

with boundary conditions

$$u = 15y \qquad \qquad x = 0, \qquad (18)$$
$$u = 25 - 18y \qquad \qquad x = 2. \qquad (19)$$

No material parameters to specify.

3.4.3 *Computational model*

- Commandline arguments are:

    float: length along x-direction
    float: length along y-direction
    float: length along z-direction (set to zero for 2D)
    integer: number of elements in x-direction
    integer: number of elements in y-direction
    integer: number of elements in z-direction (set to zero for 2D)
    integer: interpolation order (1: linear; 2: quadratic)
    integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

    2.0 1.0 0.0 2 1 0 1 0
    2.0 1.0 0.0 4 2 0 1 0
    2.0 1.0 0.0 8 4 0 1 0
    2.0 1.0 0.0 2 1 0 2 0
    2.0 1.0 0.0 4 2 0 2 0
    2.0 1.0 0.0 8 4 0 2 0
    2.0 1.0 0.0 2 1 0 1 1
    2.0 1.0 0.0 4 2 0 1 1

```
2.0 1.0 0.0 8 4 0 1 1

2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1
```

3.4.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 24 / 24
```

```
No failed tests.
```


Figure 9: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

Figure 10: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

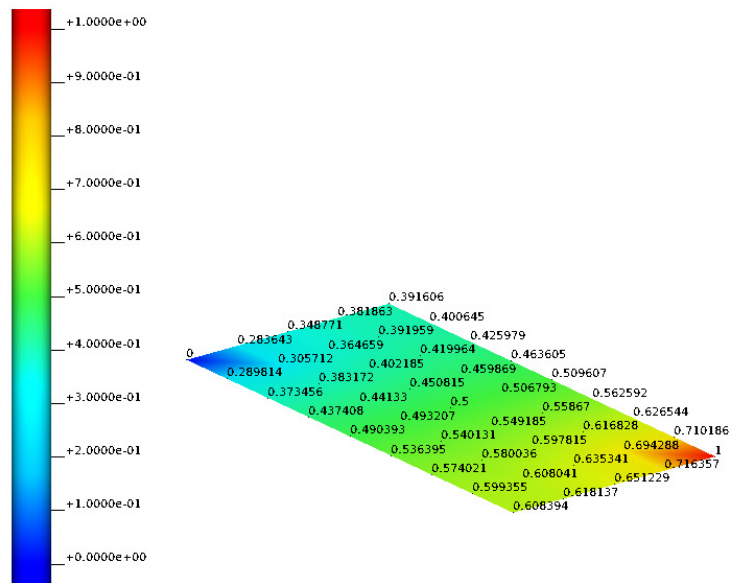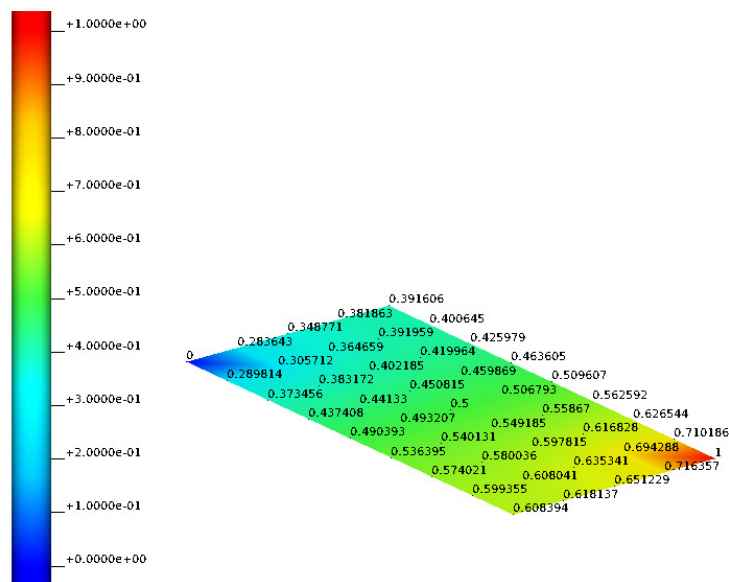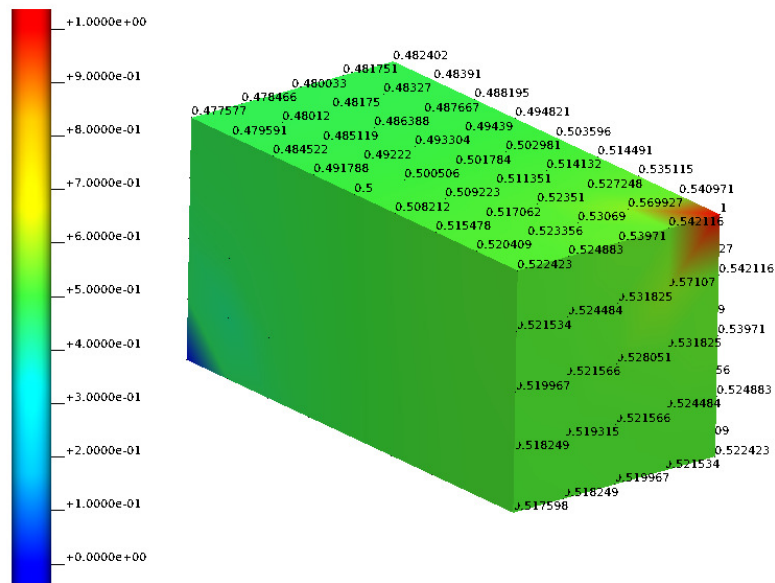**Figure 11:** 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].
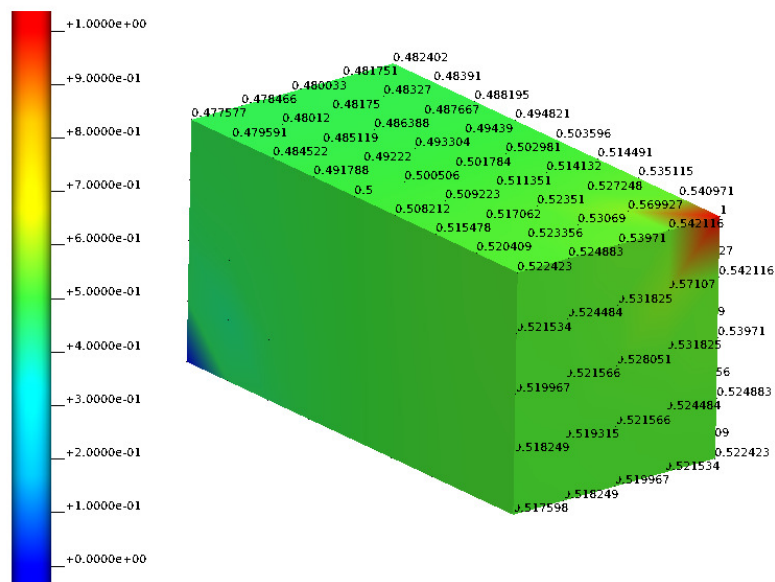
Figure 12: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

3.5   Example–0003 [`COMPILES`]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

Issues: Not clear how to prescribe Neumann BC. Results seem weird. Reference results are set up.

3.5.1   *Mathematical model – 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad\qquad \Omega = [0, 2] \times [0, 1], \tag{20}$$

with boundary conditions

$$u = 15y \qquad\qquad x = 0, \tag{21}$$
$$\partial_n u = 25 - 18y \qquad\qquad x = 2. \tag{22}$$

No material parameters to specify.

3.5.2   *Mathematical model – 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad\qquad \Omega = [0, 2] \times [0, 1] \times [0, 1], \tag{23}$$

with boundary conditions

$$u = 15y \qquad\qquad x = 0, \tag{24}$$
$$\partial_n u = 25 - 18y \qquad\qquad x = 2. \tag{25}$$

No material parameters to specify.

3.5.3   *Computational model*

- Commandline arguments are:

    float: length along x-direction

    float: length along y-direction

    float: length along z-direction (set to zero for 2D)

    integer: number of elements in x-direction

    integer: number of elements in y-direction

    integer: number of elements in z-direction (set to zero for 2D)

    integer: interpolation order (1: linear; 2: quadratic)

    integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

    2.0 1.0 0.0 2 1 0 1 0

    2.0 1.0 0.0 4 2 0 1 0

    2.0 1.0 0.0 8 4 0 1 0

    2.0 1.0 0.0 2 1 0 2 0

    2.0 1.0 0.0 4 2 0 2 0

2.0 1.0 0.0 8 4 0 2 0

2.0 1.0 0.0 2 1 0 1 1

2.0 1.0 0.0 4 2 0 1 1

2.0 1.0 0.0 8 4 0 1 1

2.0 1.0 0.0 2 1 0 2 1

2.0 1.0 0.0 4 2 0 2 1

2.0 1.0 0.0 8 4 0 2 1

2.0 1.0 1.0 2 1 1 1 0

2.0 1.0 1.0 4 2 2 1 0

2.0 1.0 1.0 8 4 4 1 0

2.0 1.0 1.0 2 1 1 2 0

2.0 1.0 1.0 4 2 2 2 0

2.0 1.0 1.0 8 4 4 2 0

2.0 1.0 1.0 2 1 1 1 1

2.0 1.0 1.0 4 2 2 1 1

2.0 1.0 1.0 8 4 4 1 1

2.0 1.0 1.0 2 1 1 2 1

2.0 1.0 1.0 4 2 2 2 1

2.0 1.0 1.0 8 4 4 2 1

3.5.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 0 / 24

Failed tests:
current_run/l2x1x0_n2x1x0_i1_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 44.2627
current_run/l2x1x0_n4x2x0_i1_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 37.2770
current_run/l2x1x0_n8x4x0_i1_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 32.2165
current_run/l2x1x0_n2x1x0_i2_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 27.3358
current_run/l2x1x0_n4x2x0_i2_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 22.1869
current_run/l2x1x0_n8x4x0_i2_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 19.7449
current_run/l2x1x0_n2x1x0_i1_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 44.2627
current_run/l2x1x0_n4x2x0_i1_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 37.2770
current_run/l2x1x0_n8x4x0_i1_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 32.2165
current_run/l2x1x0_n2x1x0_i2_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 27.3358
current_run/l2x1x0_n4x2x0_i2_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 22.1869
current_run/l2x1x0_n8x4x0_i2_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 19.7449
current_run/l2x1x1_n2x1x1_i1_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 124.749
current_run/l2x1x1_n4x2x2_i1_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 128.672
current_run/l2x1x1_n8x4x4_i1_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 143.606
current_run/l2x1x1_n2x1x1_i2_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 94.2619
current_run/l2x1x1_n4x2x2_i2_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 98.7606
current_run/l2x1x1_n8x4x4_i2_s0/Example.part0.exnode      | CHeart   - Iron |_2 = 118.047
current_run/l2x1x1_n2x1x1_i1_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 124.749
current_run/l2x1x1_n4x2x2_i1_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 128.672
current_run/l2x1x1_n8x4x4_i1_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 143.606
current_run/l2x1x1_n2x1x1_i2_s1/Example.part0.exnode      | CHeart   - Iron |_2 = 94.2619
```

```
current_run/l2x1x1_n4x2x2_i2_s1/Example.part0.exnode     | CHeart   - Iron |_2 = 98.7606
current_run/l2x1x1_n8x4x4_i2_s1/Example.part0.exnode     | CHeart   - Iron |_2 = 118.047
```
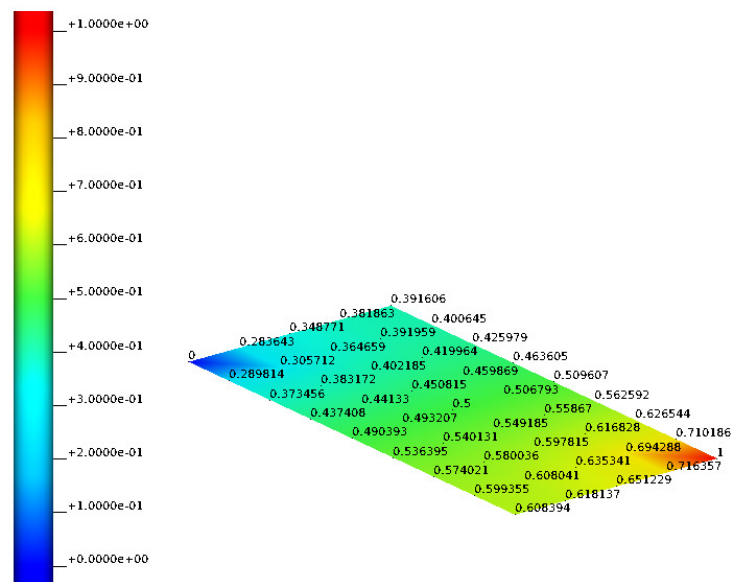
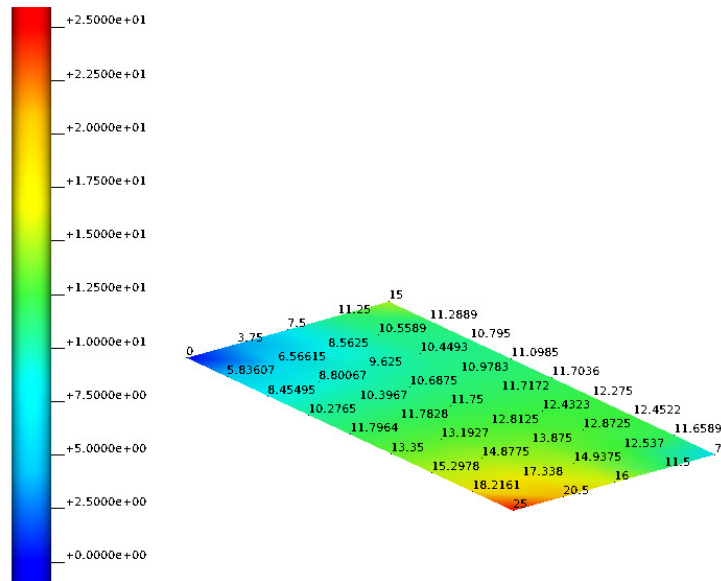**Figure 13:** 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

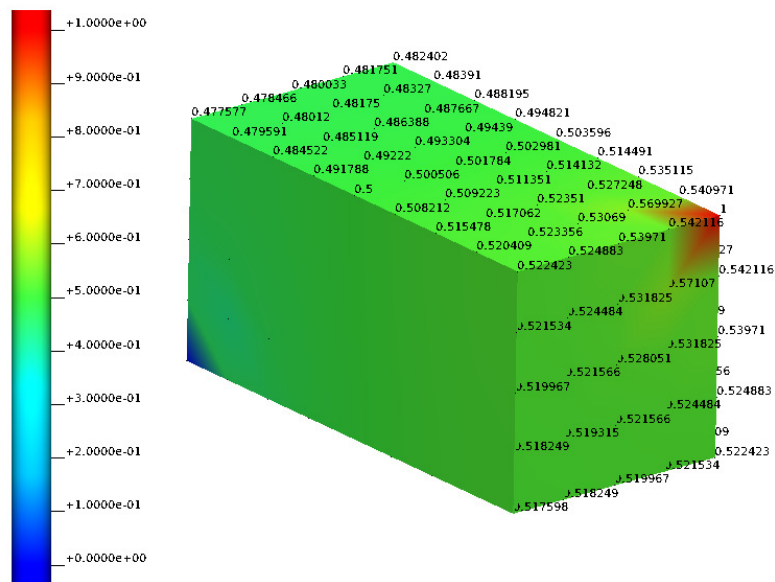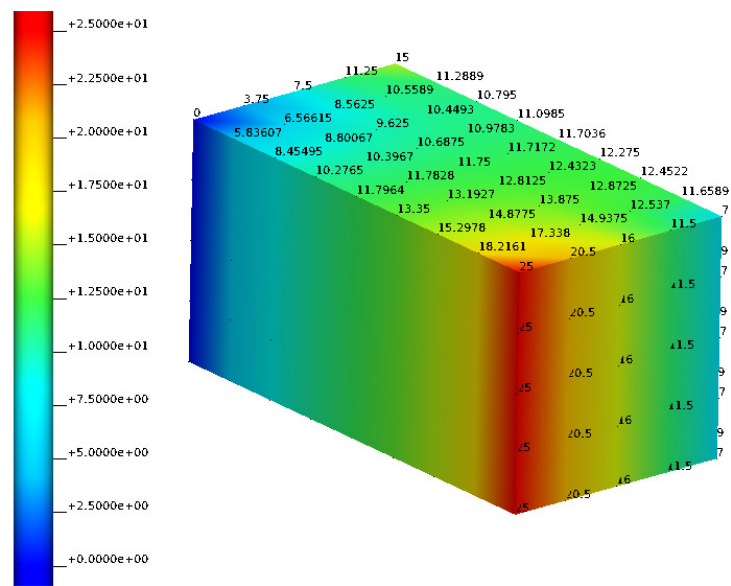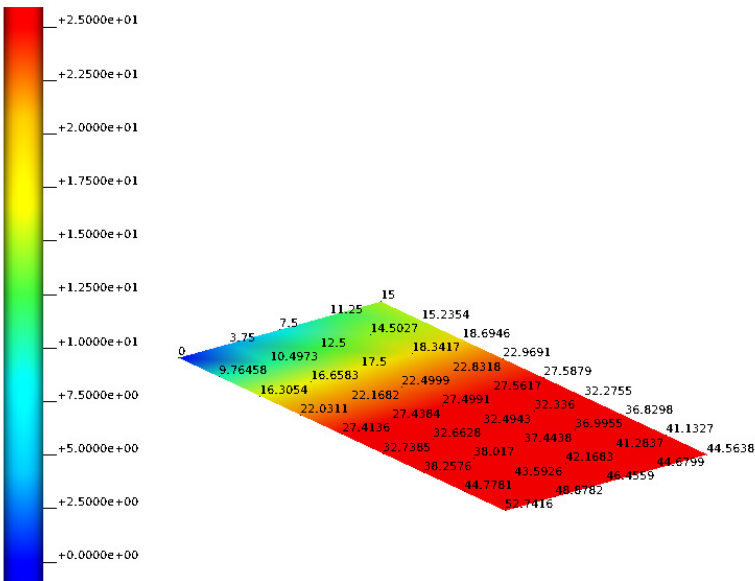**Figure 14:** 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

Figure 15: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].
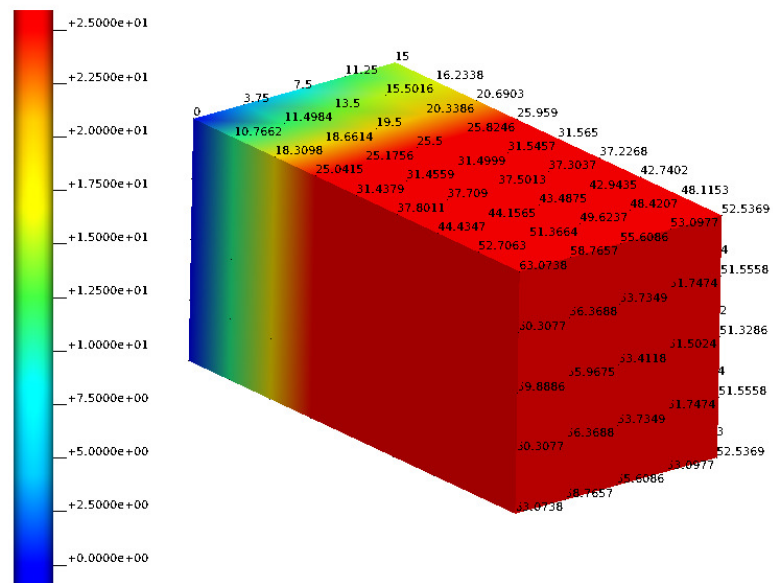
Figure 16: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

3.6   Example–0004 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

3.6.1   *Mathematical model – 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad\qquad \Omega = [0,2] \times [0,1], \qquad\qquad (26)$$

with boundary conditions

$$u = 2.0e^x \cdot \cos(y) \qquad\qquad \text{on } \partial\Omega. \qquad\qquad (27)$$

No material parameters to specify.

3.6.2   *Computational model*

- Commandline arguments are:

  integer: number of elements in x-direction

  integer: number of elements in y-direction

  integer: number of elements in z-direction (set to zero for 2D)

  integer: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

  4 2 0 1 0

  8 4 0 1 0

  2 1 0 2 0

  4 2 0 2 0

  8 4 0 2 0

  4 2 0 1 1

  8 4 0 1 1

  2 1 0 2 1

  4 2 0 2 1

  8 4 0 2 1

  100 50 0 1 0 (not tested yet..)

  100 50 0 2 0 (not tested yet..)

  100 50 0 1 1 (not tested yet..)

  100 50 0 2 1 (not tested yet..)

3.6.3   *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 10 / 10
```

```
No failed tests.
```

Figure 17: 2D results, iron reference w/ command line arguments [8 4 0 2 0].
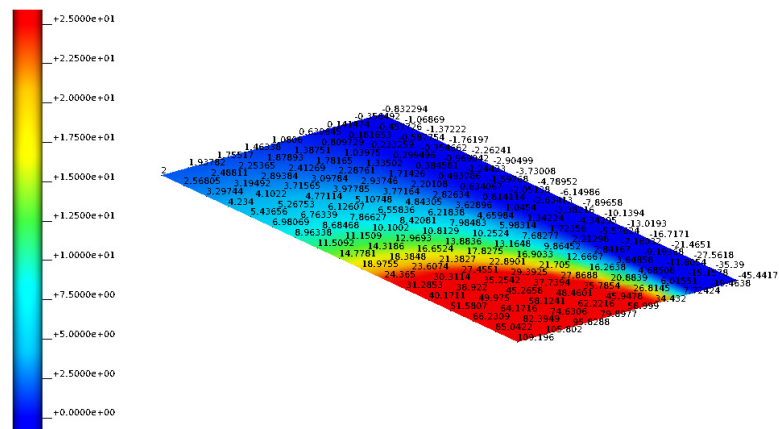
Figure 18: 2D results, current run w/ command line arguments [8 4 0 2 0].

3.7   Example-0005 [VALIDATED]

Example uses two user-defined (2d and 3d) unregular meshes and solves a patch test in form of a static problem, i.e., applies the boundary conditions in one step. Here, a Laplace problem with two Dirichlet boundary conditions is solved. The analytical solution for this setting is known and results in a linear distribution of the scalar Laplace parameter u. The numerical solution has to recover this the linear distribution in order to pass the Patch test. 2d and 3d meshes are according to MacNeil and Harder (1985).

3.7.1   *Mathematical model – 2D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad\qquad \Omega = [0, 0.24] \times [0, 0.12], \qquad (28)$$

with boundary conditions

$$u = 0 \qquad\qquad x = 0, \qquad (29)$$
$$u = 1 \qquad\qquad x = 0.24. \qquad (30)$$

No material parameters to specify.

3.7.2   *Mathematical model – 3D*

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \qquad\qquad \Omega = [0, 1] \times [0, 1] \times [0, 1], \qquad (31)$$

with boundary conditions

$$u = 0 \qquad\qquad x = 0, \qquad (32)$$
$$u = 1 \qquad\qquad x = 1. \qquad (33)$$

No material parameters to specify.

3.7.3   *Computational model*

- Commandline arguments are:

    integer: dimension (2: 2d; 3: 3d)

    integer: interpolation order (1: linear; 2: quadratic)

    integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

    2 1 0

    2 2 0

    2 1 1

    2 2 1

    3 1 0

    3 2 0

    3 1 1

    3 2 1

3.7.4 *Result summary*

Since the analytical result is known and the numerical results have to recover the linear distribution of u, the comparisons are done with the analytical solution.

```
Passed tests: 8 / 8

No failed tests.
```
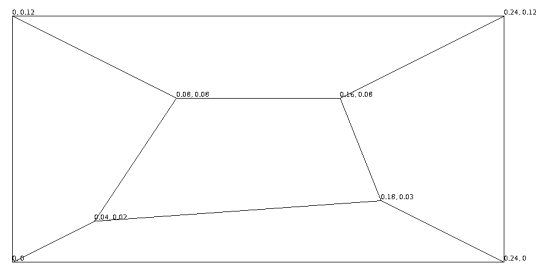

Figure 19: 2D geometry and mesh.
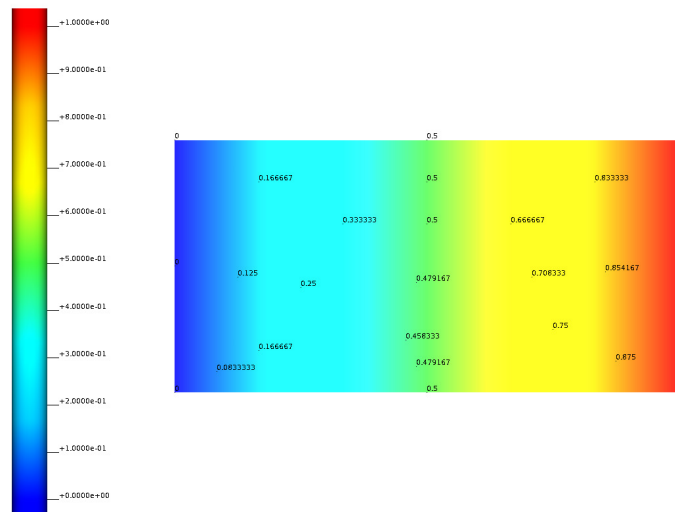
Figure 20: 2D results, iron reference w/ command line arguments [2 2 0].
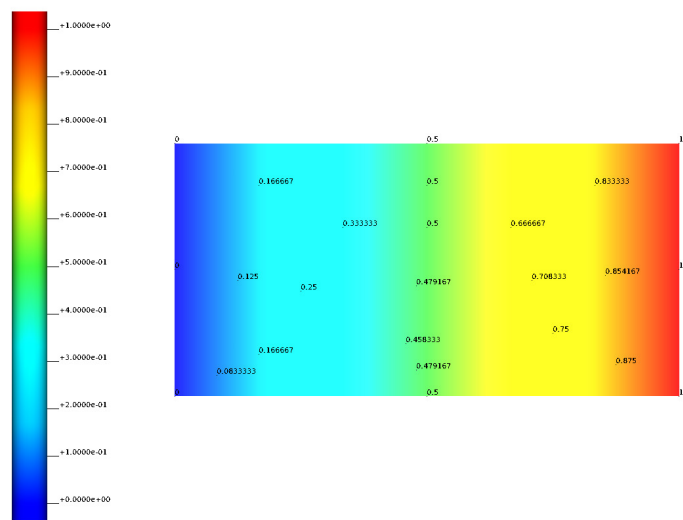
Figure 21: 2D results, current run w/ command line arguments [2 2 0].
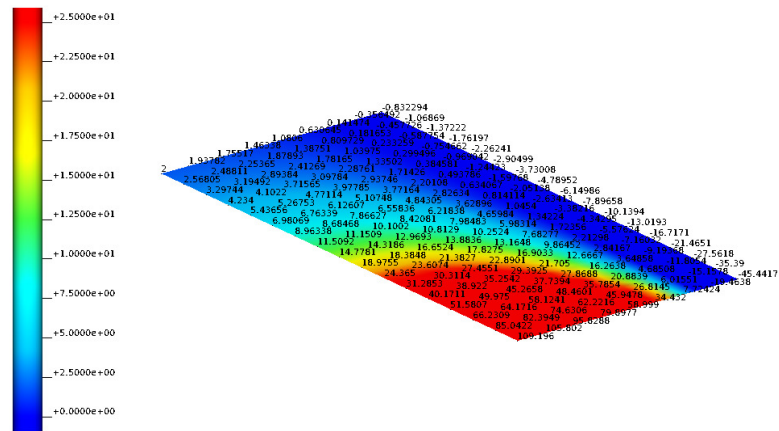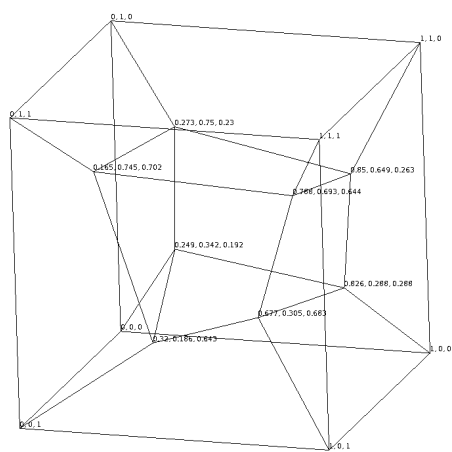

Figure 22: 3D geometry (unit cube) and mesh.
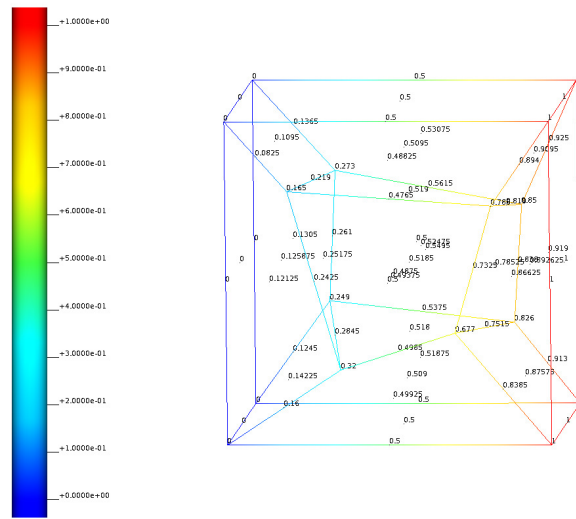
Figure 23: 3D results, iron reference w/ command line arguments [3 2 0].
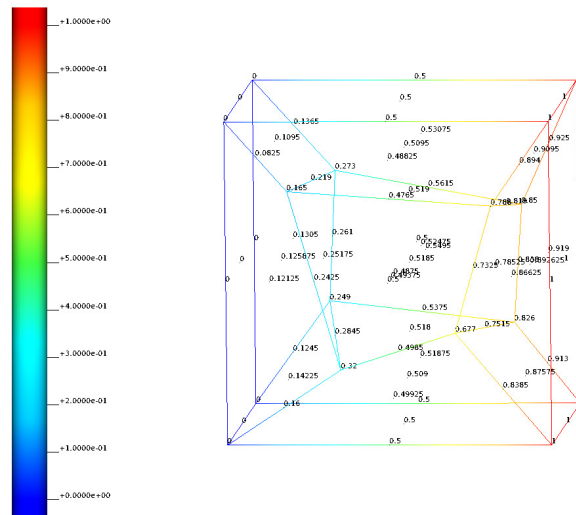
Figure 24: 3D results, current run w/ command line arguments [3 2 0].

3.8   Example–0011 [RUNS]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

3.8.1   *Mathematical model – 2D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad\qquad \Omega = [0,2] \times [0,1], \qquad (34)$$

with boundary conditions

$$u = 0 \qquad\qquad x = y = 0, \qquad (35)$$
$$u = 1 \qquad\qquad x = 2, y = 1. \qquad (36)$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \mathbf{I}. \qquad (37)$$

3.8.2   *Mathematical model – 3D*

We solve the following scalar equation,

$$\nabla \cdot [\boldsymbol{\sigma} \nabla u] = 0 \qquad\qquad \Omega = [0,2] \times [0,1] \times [0,1], \qquad (38)$$

with boundary conditions

$$u = 0 \qquad\qquad x = y = z = 0, \qquad (39)$$
$$u = 1 \qquad\qquad x = 2, y = z = 1. \qquad (40)$$

The conductivity tensor is defined as,

$$\boldsymbol{\sigma}(x,t) = \boldsymbol{\sigma} = \mathbf{I}. \qquad (41)$$

3.8.3   *Computational model*

- Commandline arguments are:

    float: length along x-direction

    float: length along y-direction

    float: length along z-direction (set to zero for 2D)

    integer: number of elements in x-direction

    integer: number of elements in y-direction

    integer: number of elements in z-direction (set to zero for 2D)

    integer: interpolation order (1: linear; 2: quadratic)

    integer: solver type (0: direct; 1: iterative)

    float: σ_{11}

    float: σ_{22}

    float: σ_{33} (ignored for 2D)

- Commandline arguments for tests are:

  2.0 1.0 0.0 2 1 0 1 0 1 1

  2.0 1.0 0.0 4 2 0 1 0 1 1

  2.0 1.0 0.0 8 4 0 1 0 1 1

  2.0 1.0 0.0 2 1 0 2 0 1 1

  2.0 1.0 0.0 4 2 0 2 0 1 1

  2.0 1.0 0.0 8 4 0 2 0 1 1

  2.0 1.0 0.0 2 1 0 1 1 1 1

  2.0 1.0 0.0 4 2 0 1 1 1 1

  2.0 1.0 0.0 8 4 0 1 1 1 1

  2.0 1.0 0.0 2 1 0 2 1 1 1

  2.0 1.0 0.0 4 2 0 2 1 1 1

  2.0 1.0 0.0 8 4 0 2 1 1 1

  2.0 1.0 1.0 2 1 1 1 0 1 1 1

  2.0 1.0 1.0 4 2 2 1 0 1 1 1

  2.0 1.0 1.0 8 4 4 1 0 1 1 1

  2.0 1.0 1.0 2 1 1 2 0 1 1 1

  2.0 1.0 1.0 4 2 2 2 0 1 1 1

  2.0 1.0 1.0 8 4 4 2 0 1 1 1

  2.0 1.0 1.0 2 1 1 1 1 1 1 1

  2.0 1.0 1.0 4 2 2 1 1 1 1 1

  2.0 1.0 1.0 8 4 4 1 1 1 1 1

  2.0 1.0 1.0 2 1 1 2 1 1 1 1

  2.0 1.0 1.0 4 2 2 2 1 1 1 1

  2.0 1.0 1.0 8 4 4 2 1 1 1 1

3.8.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

```
Passed tests: 24 / 24
```

```
No failed tests.
```

Figure 25: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1].
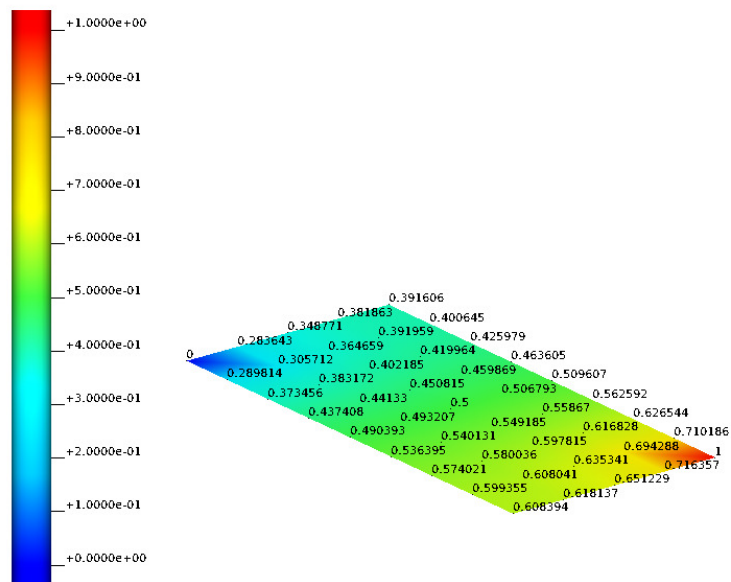
Figure 26: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1].
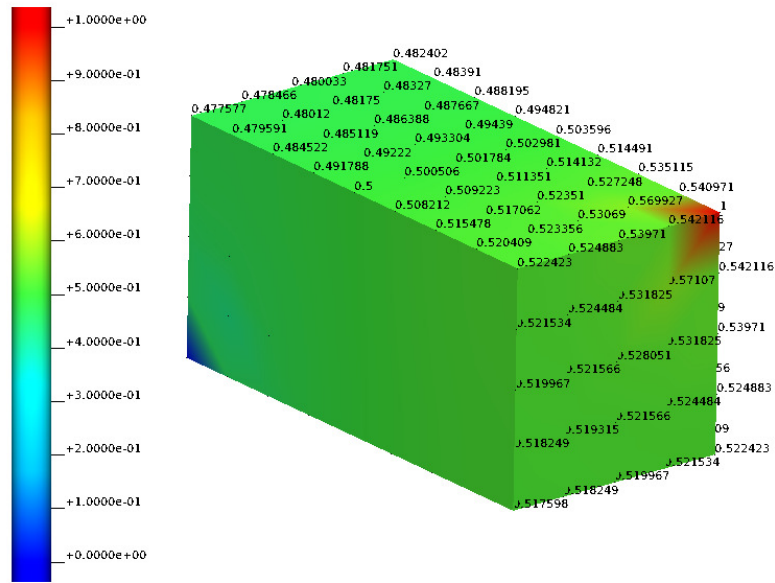
Figure 27: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 1 1 1].
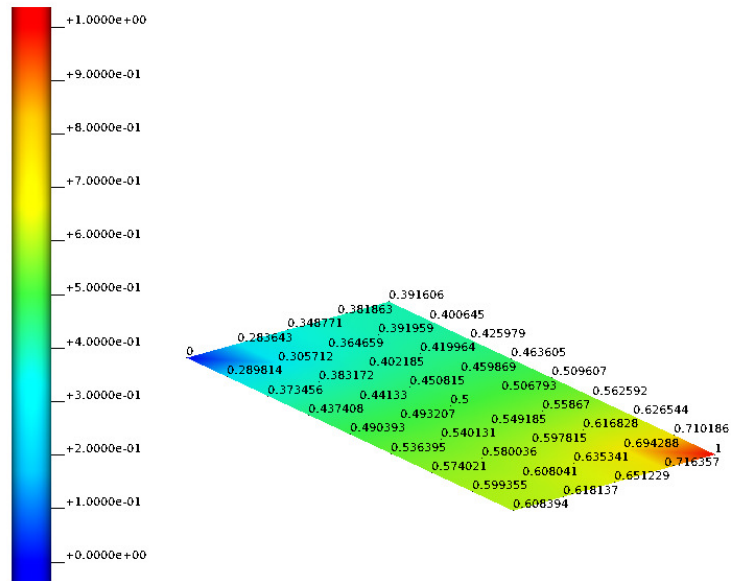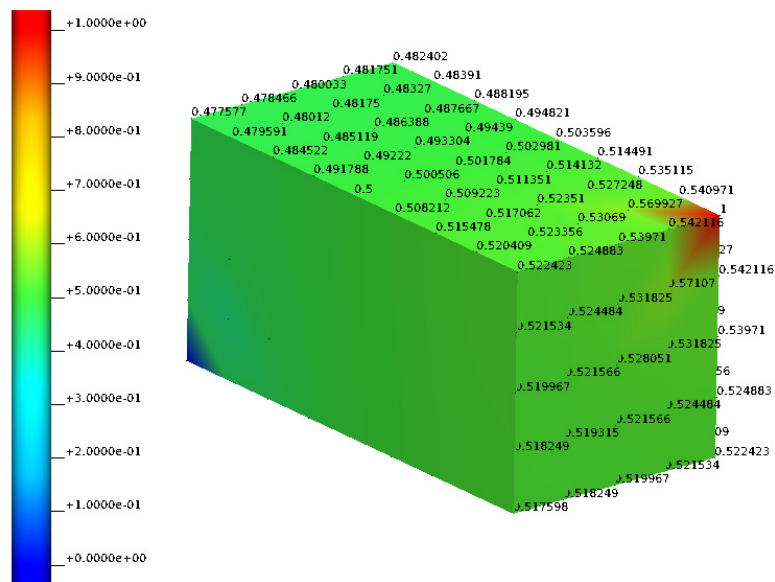

Figure 28: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 1 1 1].

4 LINEAR ELASTICITY

4.1 Equation in general form

$$\partial_{tt}\mathbf{u} + \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{f}(\mathbf{u}, t) \tag{42}$$

4.2 Example–0101 [`PLAUSIBLE`]

4.2.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{0} \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \qquad (43)$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad\qquad x = y = 0, \qquad (44)$$
$$u_x = 16 \qquad\qquad x = 160, \qquad (45)$$

and in 3D by

$$u_x = u_y = u_z = 0 \qquad\qquad x = y = z = 0, \qquad (46)$$
$$u_x = 16 \qquad\qquad x = 160. \qquad (47)$$

The material parameters are

$$E = 10000 \text{MPa}, \qquad (48)$$
$$\nu = 0.3, \qquad (49)$$
$$\rho = 5 \times 10^{-9} \text{tonne.mm}^3. \qquad (50)$$

4.2.2 *Computational model*

- Commandline arguments are:

    float: length along x-direction

    float: length along y-direction

    float: length along z-direction (set to zero for 2D)

    integer: number of elements in x-direction

    integer: number of elements in y-direction

    integer: number of elements in z-direction (set to zero for 2D)

    integer: interpolation order (1: linear; 2: quadratic)

    integer: solver type (0: direct; 1: iterative)

    float: elastic modulus

    float: Poisson ratio

    float: displacement percentage load

- Command line arguments for tests are:

    160 120 0 8 6 0 1 0 10000 0.3 0.05

    160 120 0 16 12 0 1 0 10000 0.3 0.05

    160 120 0 32 24 0 1 0 10000 0.3 0.05

    160 120 120 8 6 6 1 0 10000 0.3 0.05

    160 120 120 16 12 12 1 0 10000 0.3 0.05

    160 120 120 32 24 24 1 0 10000 0.3 0.05

    160 120 0 8 6 0 2 0 10000 0.3 0.05

    160 120 0 16 12 0 2 0 10000 0.3 0.05

160 120 0 32 24 0 2 0 10000 0.3 0.05

160 120 120 8 6 6 2 0 10000 0.3 0.05

160 120 120 16 12 12 2 0 10000 0.3 0.05

160 120 120 32 24 24 2 0 10000 0.3 0.05

4.2.3 Results

Figure 29: Results, iron 2D fine mesh.

Figure 30: Results, iron 3D fine mesh.

4.2.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_y along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accodring to

$$L_2\text{-norm} = \frac{1}{N} \times \sum_{i=1}^{N} \sqrt{\left(u^i_{y,\text{abaqus}} - u^i_{y,\text{iron}}\right)^2}, \qquad (51)$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 3.
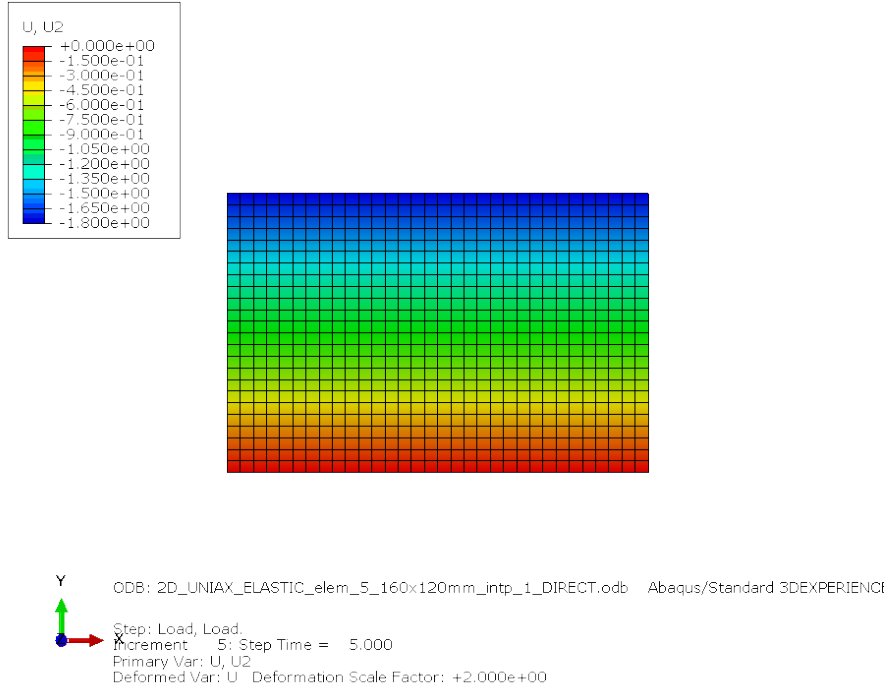
Figure 31: Results, Abaqus 2D fine mesh.
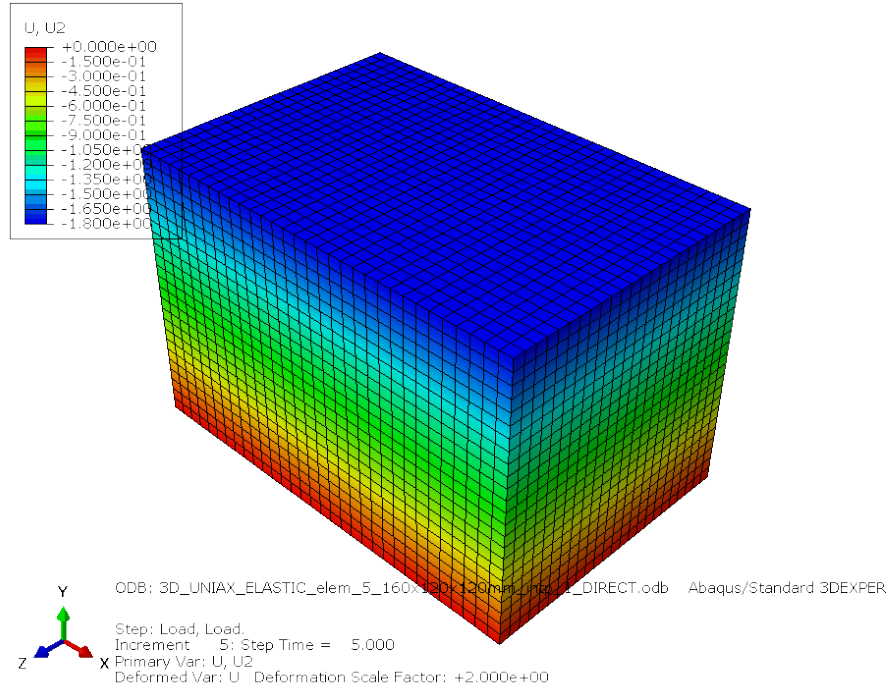
Figure 32: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-norm	Interpolation
2D	Coarse	5.322×10^{-16}	Linear
2D	Medium	1.559×10^{-15}	Linear
2D	Fine	2.900×10^{-15}	Linear
3D	Coarse	3.071×10^{-17}	Linear
3D	Medium	2.125×10^{-17}	Linear
3D	Fine	2.924×10^{-17}	Linear
2D	Coarse	9.728×10^{-16}	Quadratic
2D	Medium	2.039×10^{-15}	Quadratic
2D	Fine	2.159×10^{-15}	Quadratic
3D	Coarse	6.687×10^{-16}	Quadratic
3D	Medium	$\ldots$	Quadratic
3D	Fine	$\ldots$	Quadratic

Table 1: Quantiative error between Abaqus 2017 and iron simulations for linear elastic uniaxial extenions

4.3 Example-0102 [PLAUSIBLE]

4.3.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{0} \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \qquad (52)$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad\qquad y = 0, \qquad (53)$$
$$u_y = 8 \qquad\qquad x = 160, \qquad (54)$$

and in 3D by

$$u_x = u_z = 0 \qquad\qquad x = 0, \qquad (55)$$
$$u_y = 0 \qquad\qquad y = 0, \qquad (56)$$
$$u_x = 160 \qquad\qquad x = 160, \qquad (57)$$
$$u_y = 8 \qquad\qquad x = 160. \qquad (58)$$

The material parameters are

$$E = 10000 \text{MPa}, \qquad (59)$$
$$\nu = 0.3, \qquad (60)$$
$$\rho = 5 \times 10^{-9} \text{tonne.mm}^3. \qquad (61)$$

4.3.2 *Computational model*

- Commandline arguments are:

    float: length along x-direction

    float: length along y-direction

    float: length along z-direction (set to zero for 2D)

    integer: number of elements in x-direction

    integer: number of elements in y-direction

    integer: number of elements in z-direction (set to zero for 2D)

    integer: interpolation order (1: linear; 2: quadratic)

    integer: solver type (0: direct; 1: iterative)

    float: elastic modulus

    float: Poisson ratio

    float: displacement percentage load

- Command line arguments for tests are:

    160 120 0 8 6 0 1 0 10000 0.3 0.05

    160 120 0 16 12 0 1 0 10000 0.3 0.05

    160 120 0 32 24 0 1 0 10000 0.3 0.05

    160 120 120 8 6 6 1 0 10000 0.3 0.05

    160 120 120 16 12 12 1 0 10000 0.3 0.05

    160 120 120 32 24 24 1 0 10000 0.3 0.05

160 120 0 8 6 0 2 0 10000 0.3 0.05

160 120 0 16 12 0 2 0 10000 0.3 0.05

160 120 0 32 24 0 2 0 10000 0.3 0.05

160 120 120 8 6 6 2 0 10000 0.3 0.05

160 120 120 16 12 12 2 0 10000 0.3 0.05

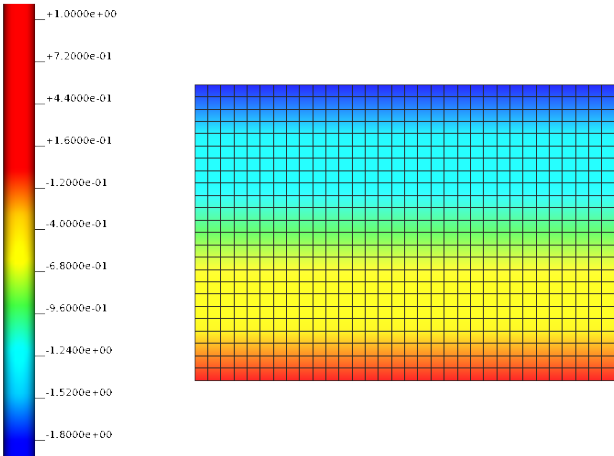160 120 120 32 24 24 2 0 10000 0.3 0.05

4.3.3   *Results*
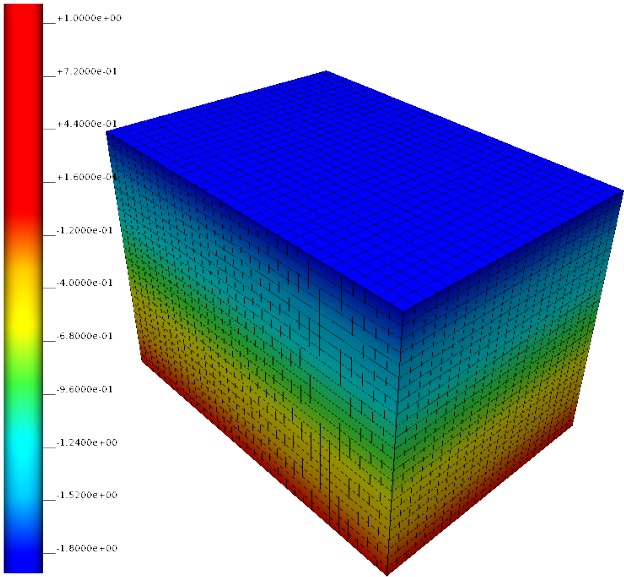

Figure 33: Results, iron 2D fine mesh.

Figure 34: Results, iron 3D fine mesh.

4.3.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_x along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accodring to

$$L_2\text{-norm} = \frac{1}{N} \times \sum_{i=1}^{N} \sqrt{\left(u_{y,\text{abaqus}}^i - u_{y,\text{iron}}^i\right)^2},\qquad (62)$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 3.
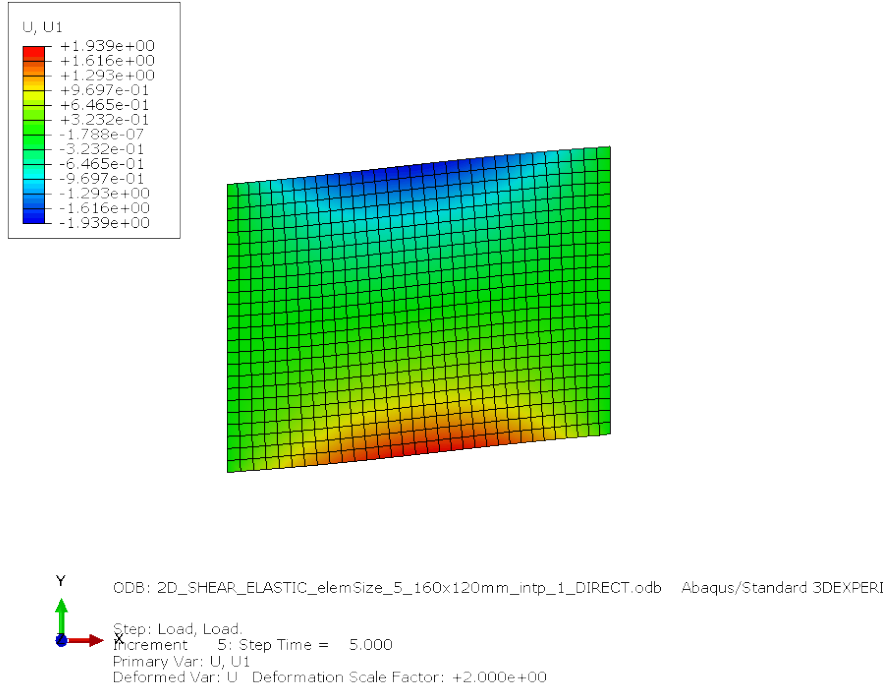
Figure 35: Results, Abaqus 2D fine mesh.
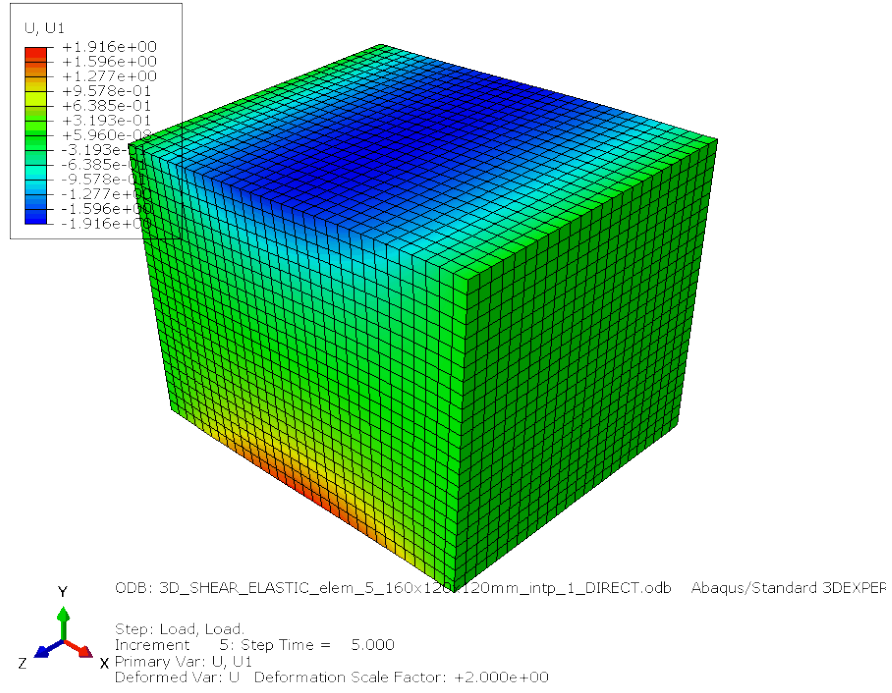
Figure 36: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-norm	Interpolation
2D	Coarse	6.696×10^{-3}	Linear
2D	Medium	1.273×10^{-3}	Linear
2D	Fine	2.489×10^{-4}	Linear
3D	Coarse	4.234×10^{-4}	Linear
3D	Medium	4.184×10^{-5}	Linear
3D	Fine	3.781×10^{-6}	Linear
2D	Coarse	3.036×10^{-4}	Quadratic
2D	Medium	6.099×10^{-5}	Quadratic
2D	Fine	1.089×10^{-5}	Quadratic
3D	Coarse	. . .	Quadratic
3D	Medium	. . .	Quadratic
3D	Fine	. . .	Quadratic

Table 2: Quantiative error between Abaqus 2017 and iron simulations for linear elastic shear

4.4 Example–0111 [PLAUSIBLE]

4.4.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{f}(\mathbf{u}, t) \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \qquad (63)$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad\qquad x = y = 0, \qquad (64)$$
$$\mathbf{f}(u_x) = 6.0 \times 10^4 \qquad\qquad x = 160, \qquad (65)$$

and in 3D by

$$u_x = u_y = u_z = 0 \qquad\qquad x = y = z = 0, \qquad (66)$$
$$\mathbf{f}(u_x) = 7.2 \times 10^6 \qquad\qquad x = 160. \qquad (67)$$

The material parameters are

$$E = 10000 \text{MPa}, \qquad (68)$$
$$\nu = 0.3, \qquad (69)$$
$$\rho = 5 \times 10^{-9} \text{tonne.mm}^3. \qquad (70)$$

4.4.2 *Computational model*

- Commandline arguments are:

    float: length along x-direction

    float: length along y-direction

    float: length along z-direction (set to zero for 2D)

    integer: number of elements in x-direction

    integer: number of elements in y-direction

    integer: number of elements in z-direction (set to zero for 2D)

    integer: interpolation order (1: linear; 2: quadratic)

    integer: solver type (0: direct; 1: iterative)

    float: elastic modulus

    float: Poisson ratio

    float: XXX

- Command line arguments for tests are:

    160 120 0 8 6 0 1 0 10000 0.3 XXX

    160 120 0 16 12 0 1 0 10000 0.3 XXX

    160 120 0 32 24 0 1 0 10000 0.3 XXX

    160 120 120 8 6 6 1 0 10000 0.3 XXX

    160 120 120 16 12 12 1 0 10000 0.3 XXX

    160 120 120 32 24 24 1 0 10000 0.3 XXX

    160 120 0 8 6 0 2 0 10000 0.3 XXX

    160 120 0 16 12 0 2 0 10000 0.3 XXX

160 120 0 32 24 0 2 0 10000 0.3 XXX

160 120 120 8 6 6 2 0 10000 0.3 XXX

160 120 120 16 12 12 2 0 10000 0.3 XXX

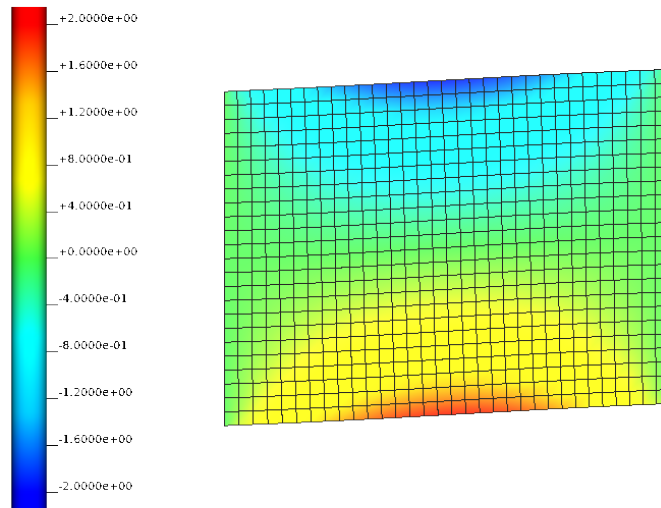160 120 120 32 24 24 2 0 10000 0.3 XXX

4.4.3 Results
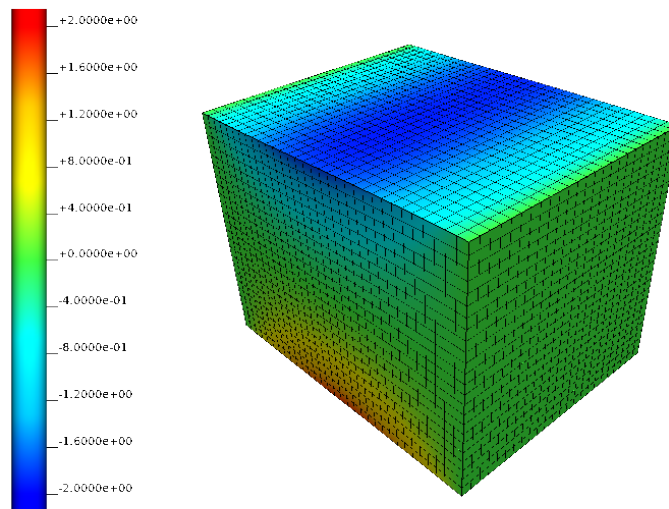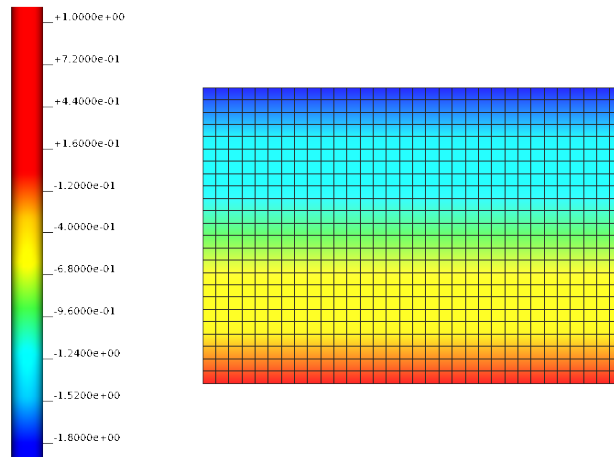

Figure 37: Results, iron 2D fine mesh.

Figure 38: Results, iron 3D fine mesh.

4.4.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_y along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accodring to

$$L_2\text{-norm} = \frac{1}{N} \times \sum_{i=1}^{N} \sqrt{\left(u_{y,\mathrm{abaqus}}^{i} - u_{y,\mathrm{iron}}^{i}\right)^2}, \tag{71}$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 3.
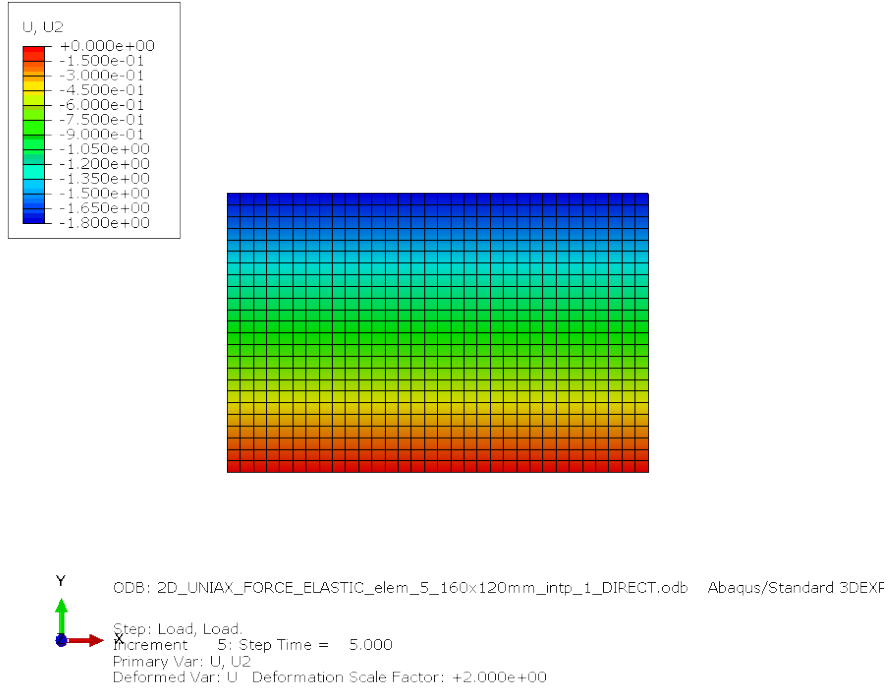
Figure 39: Results, Abaqus 2D fine mesh.
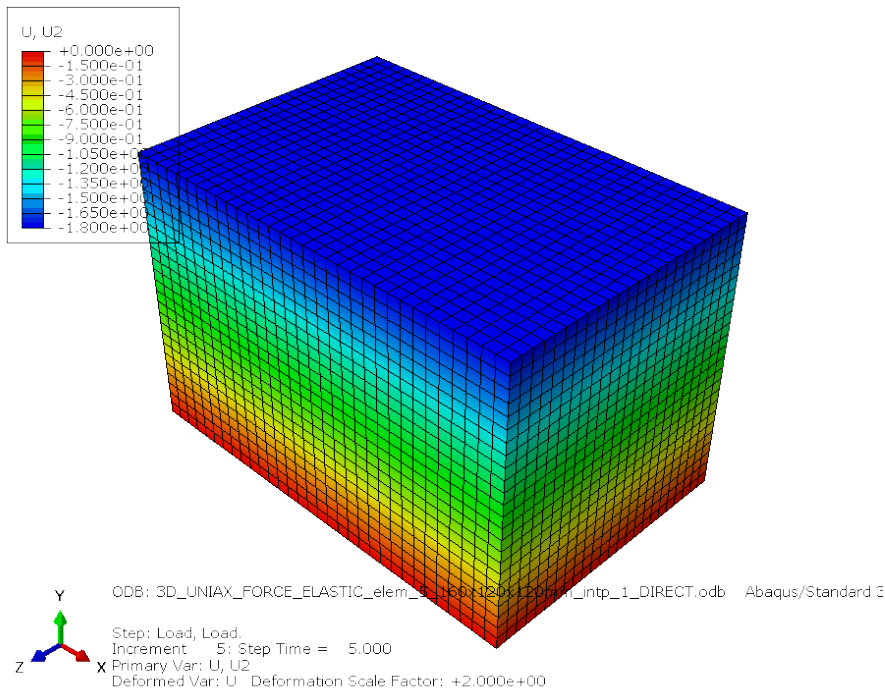
Figure 40: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-norm	Interpolation
2D	Coarse	...	Linear
2D	Medium	...	Linear
2D	Fine	...	Linear
3D	Coarse	...	Linear
3D	Medium	...	Linear
3D	Fine	...	Linear
2D	Coarse	...	Quadratic
2D	Medium	...	Quadratic
2D	Fine	...	Quadratic
3D	Coarse	...	Quadratic
3D	Medium	...	Quadratic
3D	Fine	...	Quadratic

Table 3: Quantiative error between Abaqus 2017 and iron simulations for linear elastic uniaxial extenions

4.5   Example-0112 [PLAUSIBLE]

4.5.1 *Mathematical model*

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{f}(\mathbf{u}, t) \qquad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \qquad (72)$$

with time step size $\Delta_t = 1$ and $\mathbf{u} = [u_x, u_y]$ in 2D $\mathbf{u} = [u_x, u_y, u_z]$ in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \qquad\qquad y = 0, \qquad (73)$$
$$\mathbf{f}(u_y) = 6.0 \times 10^4 \qquad\qquad x = 160, \qquad (74)$$

and in 3D by

$$u_x = u_z = 0 \qquad\qquad x = 0, \qquad (75)$$
$$u_y = 0 \qquad\qquad y = 0, \qquad (76)$$
$$u_x = 160 \qquad\qquad x = 160, \qquad (77)$$
$$\mathbf{f}(u_y) = 7.2 \times 10^6 \qquad\qquad x = 160. \qquad (78)$$

The material parameters are

$$E = 10000 \text{MPa}, \qquad (79)$$
$$\nu = 0.3, \qquad (80)$$
$$\rho = 5 \times 10^{-9} \text{tonne.mm}^3. \qquad (81)$$

4.5.2 *Computational model*

- Commandline arguments are:

  float: length along x-direction

  float: length along y-direction

  float: length along z-direction (set to zero for 2D)

  integer: number of elements in x-direction

  integer: number of elements in y-direction

  integer: number of elements in z-direction (set to zero for 2D)

  integer: interpolation order (1: linear; 2: quadratic)

  integer: solver type (0: direct; 1: iterative)

  float: elastic modulus

  float: Poisson ratio

  float: XXX

- Command line arguments for tests are:

  160 120 0 8 6 0 1 0 10000 0.3 XXX

  160 120 0 16 12 0 1 0 10000 0.3 XXX

  160 120 0 32 24 0 1 0 10000 0.3 XXX

  160 120 120 8 6 6 1 0 10000 0.3 XXX

  160 120 120 16 12 12 1 0 10000 0.3 XXX

  160 120 120 32 24 24 1 0 10000 0.3 XXX

160 120 0 8 6 0 2 0 10000 0.3 XXX

160 120 0 16 12 0 2 0 10000 0.3 XXX

160 120 0 32 24 0 2 0 10000 0.3 XXX

160 120 120 8 6 6 2 0 10000 0.3 XXX

160 120 120 16 12 12 2 0 10000 0.3 XXX

160 120 120 32 24 24 2 0 10000 0.3 XXX

4.5.3 *Results*
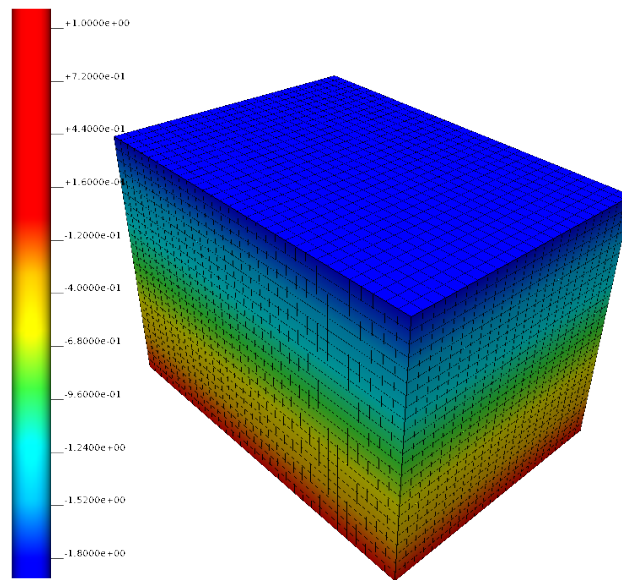

Figure 41: Results, iron 2D fine mesh.
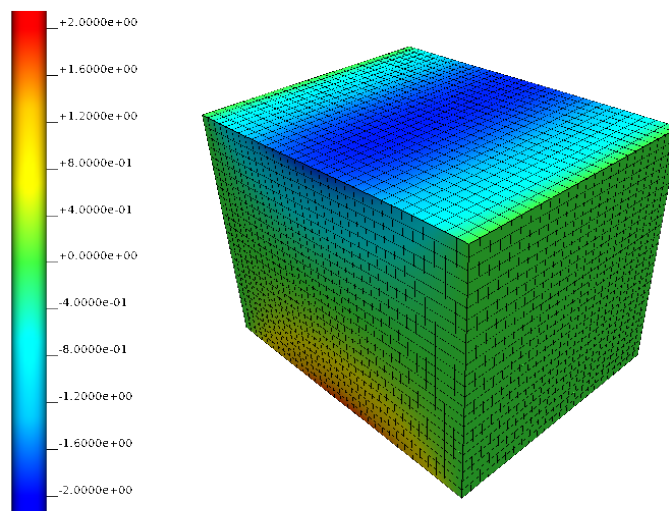
Figure 42: Results, iron 3D fine mesh.

4.5.4 *Validation*

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by comparing the horizontal displacement u_x along the free-edge ($y = 120$ for 2D and $y = z = 120$ for 3D) and computing the L2-norm accodring to

$$\mathtt{L_2\text{-}norm} = \frac{1}{N} \times \sum_{i=1}^{N} \sqrt{\left(u^i_{y,\mathtt{abaqus}} - u^i_{y,\mathtt{iron}} \right)^2}, \tag{82}$$

where N is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 3.
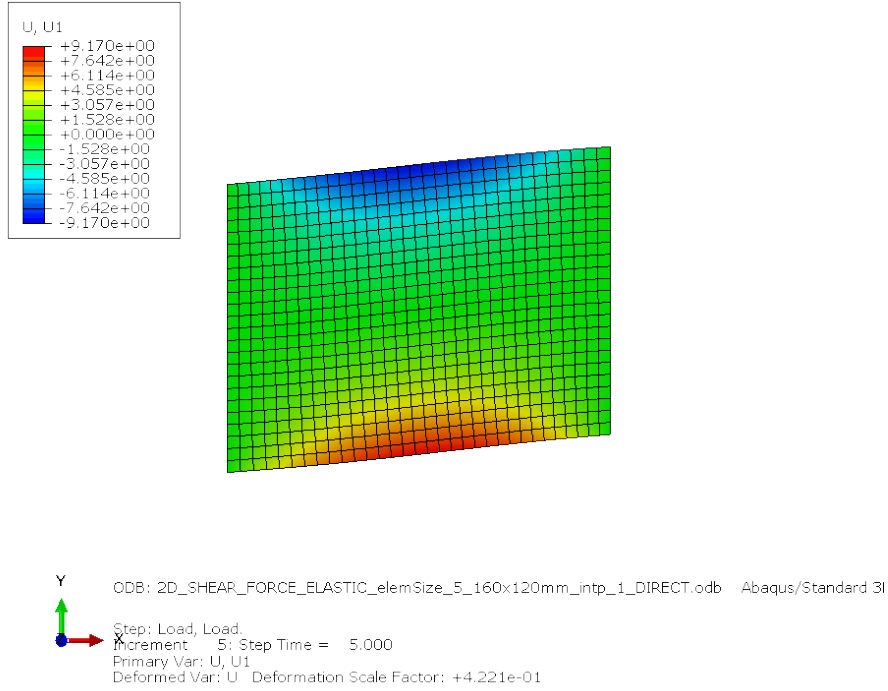
Figure 43: Results, Abaqus 2D fine mesh.
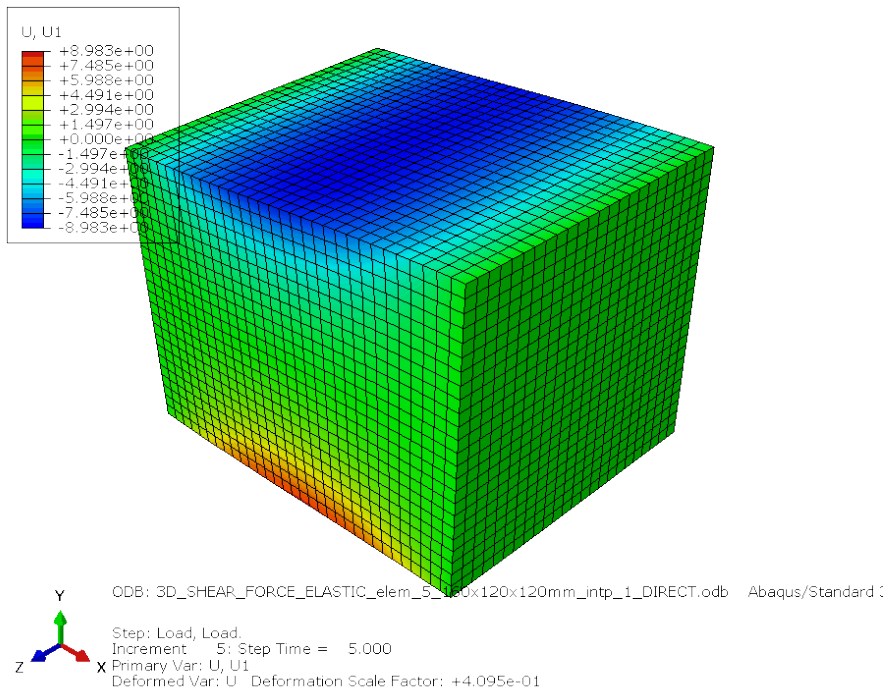
Figure 44: Results, abaqus 3D fine mesh.

Dimension	Mesh	L_2-norm	Interpolation
2D	Coarse	...	Linear
2D	Medium	...	Linear
2D	Fine	...	Linear
3D	Coarse	...	Linear
3D	Medium	...	Linear
3D	Fine	...	Linear
2D	Coarse	...	Quadratic
2D	Medium	...	Quadratic
2D	Fine	...	Quadratic
3D	Coarse	...	Quadratic
3D	Medium	...	Quadratic
3D	Fine	...	Quadratic

Table 4: Quantiative error between Abaqus 2017 and iron simulations for linear elastic shear

5   FINITE ELASTICITY

6 NAVIER–STOKES FLOW

6.1 Equation in general form

$$\partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + p\mathbf{I}) = \rho \mathbf{f} \tag{83}$$

6 NAVIER–STOKES FLOW

$$\partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + p\mathbf{I}) = \rho \mathbf{f} \tag{83}$$

6.2   Example–0302–u `[COMPILES]`

Example uses user-defined simplex meshes in CHeart mesh format with quadratic/linear interpolation for velocity/pressure and solves a dynamic problem.

Setup is the well-known lid-driven cavity problem on the unit square or unit cube in two and three dimensions.

Current issue: does not converge after 30 some time iterations (2D and 3D).

Visualization issue: In exelem-file, replace

```
1.   constant(2)*constant, no modify, grid based.
   #xi1=0, #xi2=0
2.   constant(2)*constant, no modify, grid based.
```

with

```
1.   constant*constant, no modify, grid based.
   #xi1=0, #xi2=0
2.   constant*constant, no modify, grid based.
```

and likewise for 3D, replace

```
1.   constant(2;3)*constant*constant, no modify, grid based.
   #xi1=0, #xi2=0, #xi3=0
2.   constant(2;3)*constant*constant, no modify, grid based.
```

with

```
1.   constant*constant*constant, no modify, grid based.
   #xi1=0, #xi2=0, #xi3=0
2.   constant*constant*constant, no modify, grid based.
```

6.2.1   *Mathematical model – 2D*

We solve the incompressible Navier-Stokes equation,

$$\partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v}) - \nabla \cdot (\mu \nabla \boldsymbol{v} - p \mathbf{I}) = \rho \mathbf{f} \qquad \Omega = [0,1] \times [0,1], \qquad (84)$$

$$\nabla \cdot \boldsymbol{v} = 0, \qquad (85)$$

with boundary conditions

$$\boldsymbol{v} = 0 \qquad\qquad x = 0, \qquad (86)$$
$$\boldsymbol{v} = 0 \qquad\qquad x = 1, \qquad (87)$$
$$\boldsymbol{v} = 0 \qquad\qquad y = 0, \qquad (88)$$
$$\boldsymbol{v} = [1,0]^{\mathsf{T}} \qquad\qquad y = 1. \qquad (89)$$

Viscosity $\mu = 0.0025$, density $\rho = 1$. Thus, Reynolds number $Re = 400$.

6.2.2   *Mathematical model – 3D*

We solve the incompressible Navier-Stokes equation,

$$\partial_t(\rho\boldsymbol{v}) + \nabla \cdot (\rho\boldsymbol{v} \otimes \boldsymbol{v}) - \nabla \cdot (\mu\nabla\boldsymbol{v} - p\mathbf{I}) = \rho\mathbf{f} \quad \Omega = [0,1] \times [0,1] \times [0,1], \quad (90)$$

$$\nabla \cdot \boldsymbol{v} = 0, \quad (91)$$

with boundary conditions

$$\boldsymbol{v} = 0 \qquad\qquad x = 0, \qquad (92)$$
$$\boldsymbol{v} = 0 \qquad\qquad x = 1, \qquad (93)$$
$$\boldsymbol{v} = 0 \qquad\qquad y = 0, \qquad (94)$$
$$\boldsymbol{v} = [1,0]^\mathsf{T} \qquad\qquad y = 1, \qquad (95)$$
$$\boldsymbol{v} = 0 \qquad\qquad z = 0, \qquad (96)$$
$$\boldsymbol{v} = 0 \qquad\qquad z = 1. \qquad (97)$$

Viscosity $\mu = 0.01$, density $\rho = 1$. Thus, Reynolds number $\mathrm{Re} = 100$.

6.2.3   *Computational model*

- Commandline arguments are:

    integer: number of dimensions (2: 2D, 3: 3D

    integer: mesh refinement level (1, 2, 3, ...)

    float: start time

    float: stop time

    float: time step size

    float: density

    float: viscosity

    integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

    2 1 0.0 1.0 0.001 0.0025 1.0 0

    2 2 0.0 1.0 0.001 0.0025 1.0 0

    2 3 0.0 1.0 0.001 0.0025 1.0 0

    2 1 0.0 1.0 0.001 0.0025 1.0 1

    2 2 0.0 1.0 0.001 0.0025 1.0 1

    2 3 0.0 1.0 0.001 0.0025 1.0 1

    3 1 0.0 1.0 0.001 0.01 1.0 0

    3 2 0.0 1.0 0.001 0.01 1.0 0

    3 3 0.0 1.0 0.001 0.01 1.0 0

    3 1 0.0 1.0 0.001 0.01 1.0 1

    3 2 0.0 1.0 0.001 0.01 1.0 1

    3 3 0.0 1.0 0.001 0.01 1.0 1

- Note: Binary uses command line arguments to search for the relevant mesh files.

6.2.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

Passed tests: 0 / 12

All tests failed.

7   MONODOMAIN

8 CELLML MODEL

REFERENCES

[1] Chris Bradley, Andy Bowery, Randall Britten, Vincent Budelmann, Oscar Camara, Richard Christie, Andrew Cookson, Alejandro F Frangi, Thiranja Babarenda Gamage, Thomas Heidlauf, et al. Opencmiss: a multi-physics & multi-scale computational infrastructure for the vph/physiome project. *Progress in biophysics and molecular biology*, 107(1):32–47, 2011.